Search Session History

Edit an existing query or compose a new query in the Search Query Display.

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

Fri, 10 Mar 2006, 11:10:42 AM EST

Search Query Display

Run Search Reset

Recent Search Queries

#1	(((compress* <or> encod*) <and> (instruction* <near/2> address*) <and> (tracing <or> trace*))<in>metadata)
#2	(((compress* <or> encod*) <and> (instruction* <near/2> address*) <and> (tracing <or> trace*))<in>metadata)
#3	(((append* <or> prepend* <or> pre-pend*)<and> (instruction* <near/2> address*) <and> (tracing <or> trace*))<in>metadata)
#4	((((append* <or> prepend* <or> pre?pend*)<and> (instruction* <near/2> address*) <and> (isa <or> (instruction <near/1> set))))<in>metadata)

Clear Session History

Terms used **trac instruction address append prepend** Found **11,970** of **171,143**

Sort results by relevance
Display results expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown
Relevance scale

1 The family of concurrent logic programming languages
Ehud Shapiro
September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3
Publisher: ACM Press
Full text available: pdf(9.62 MB) Additional Information: full citation, abstract, references, citings, index terms

Concurrent logic languages are high-level programming languages for parallel and distributed systems that offer a wide range of both known and novel concurrent programming techniques. Being logic programming languages, they preserve many advantages of the abstract logic programming model, including the logical reading of programs and computations, the convenience of representing data structures with logical terms and manipulating them using unification, and the amenability to metaprogrammin ...

2 A trace-based evaluation of adaptive error correction for a wireless local area network
David A. Eckhardt, Peter Steenkiste
December 1999 **Mobile Networks and Applications**, Volume 4 Issue 4
Publisher: Kluwer Academic Publishers
Full text available: pdf(243.29 KB) Additional Information: full citation, abstract, references, citings, index terms

Wireless transmissions are highly susceptible to noise and interference. As a result, the error characteristics of a wireless link may vary widely depending on environmental factors such as location of the communicating systems and activity of competing radiation sources, making error control a difficult task. In this paper we evaluate error control strategies for a wireless LAN. Based on low-level packet traces of WaveLAN, we first show that forward error correction (FEC) is effective in r ...

3 A type system for Java bytecode subroutines
Raymie Stata, Martin Abadi
January 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 1
Publisher: ACM Press
Full text available: pdf(519.84 KB) Additional Information: full citation, abstract, references, citings, index terms

Java is typically compiled into an intermediate language, JVML, that is interpreted by the

Java Virtual Machine. Because mobile JVML code is not always trusted, a bytecode verifier enforces static constraints that prevent various dynamic errors. Given the importance of the bytecode verifier for security, its current descriptions are inadequate. This article proposes using typing rules to describe the bytecode verifier because they are more precise than prose, clearer than code, and easier ...

Keywords: Java, bytecode verification

4 Shangri-La: achieving high performance from compiled network applications while enabling ease of programming

Michael K. Chen, Xiao Feng Li, Ruiqi Lian, Jason H. Lin, Lixia Liu, Tao Liu, Roy Ju

June 2005 **ACM SIGPLAN Notices , Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation PLDI '05**, Volume 40 Issue 6

Publisher: ACM Press

Full text available: pdf(480.93 KB) Additional Information: full citation, abstract, references, citings, index terms

Programming network processors is challenging. To sustain high line rates, network processors have extremely tight memory access and instruction budgets. Achieving desired performance has traditionally required hand-coded assembly. Researchers have recently proposed high-level programming languages for packet processing, but the challenges of compiling these languages into code that is competitive with hand-tuned assembly remain unanswered.This paper describes the Shangri-La compiler, which acce ...

Keywords: chip multiprocessors, dataflow programming, network processors, packet processing, program partitioning, throughput-oriented computing

5 Bisimulation can't be traced

Bard Bloom, Sorin Istrail, Albert R. Meyer

January 1995 **Journal of the ACM (JACM)**, Volume 42 Issue 1

Publisher: ACM Press

Full text available: pdf(2.33 MB) Additional Information: full citation, abstract, references, citings, index terms

In the concurrent language CCS, two programs are considered the same if they are bisimilar. Several years and many researchers have demonstrated that the theory of bisimulation is mathematically appealing and useful in practice. However, bisimulation makes too many distinctions between programs. We consider the problem of adding operations to CCS to make bisimulation fully abstract. We define the class of GSOS operations, generalizing the style and technical advantages of C ...

Keywords: CCS, bisimulation, process algebra, structural operational semantics

6 C and tcc: a language and compiler for dynamic code generation

Massimiliano Poletto, Wilson C. Hsieh, Dawson R. Engler, M. Frans Kaashoek

March 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 2

Publisher: ACM Press

Full text available: pdf(471.68 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Dynamic code generation allows programmers to use run-time information in order to achieve performance and expressiveness superior to those of static code. The 'C(Tick C)

language is a superset of ANSI C that supports efficient and high-level use of dynamic code generation. 'C provides dynamic code generation at the level of C expressions and statements and supports the composition of dynamic code at run time. These features enable programmers to add dynamic code generation ...

Keywords: ANSI C, compilers, dynamic code generation, dynamic code optimization

7 tcc: a system for fast, flexible, and high-level dynamic code generation

Massimiliano Poletto, Dawson R. Engler, M. Frans Kaashoek

May 1997 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conference on Programming language design and implementation PLDI '97**, Volume 32 Issue 5

Publisher: ACM Press

Full text available: pdf(1.94 MB) Additional Information: full citation, abstract, references, citings, index terms

tcc is a compiler that provides efficient and high-level access to dynamic code generation. It implements the 'C ("Tick-C") programming language, an extension of ANSI C that supports dynamic code generation [15]. 'C gives power and flexibility in specifying dynamically generated code: whereas most other systems use annotations to denote run-time invariants. 'C allows the programmer to specify and compose arbitrary expressions and statements at run time. This degree of control is needed to effici ...

8 Computer security: Checking security of Java bytecode by abstract interpretation

Roberto Barbuti, Cinzia Bernardeschi, Nicoletta De Francesco

March 2002 **Proceedings of the 2002 ACM symposium on Applied computing**

Publisher: ACM Press

Full text available: pdf(710.13 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a method to certify a subset of the Java bytecode, with respect to security. The method is based on abstract interpretation of the operational semantics of the language. We define a concrete small-step enhanced semantics of the language, able to keep information on the flow of data and control during execution. A main point of this semantics is the handling of the influence of the information flow on the operand stack. We then define an abstract semantics, keeping only the security in ...

Keywords: Java bytecode, abstract interpretation, information flow, security

9 EROS: a fast capability system

Jonathan S. Shapiro, Jonathan M. Smith, David J. Farber

December 1999 **ACM SIGOPS Operating Systems Review , Proceedings of the seventeenth ACM symposium on Operating systems principles SOSP '99**, Volume 33 Issue 5

Publisher: ACM Press

Full text available: pdf(1.83 MB) Additional Information: full citation, abstract, references, citings, index terms

EROS is a capability-based operating system for commodity processors which uses a single level storage model. The single level store's persistence is transparent to applications. The performance consequences of support for transparent persistence and capability-based architectures are generally believed to be negative. Surprisingly, the basic operations of EROS (such as IPC) are generally comparable in cost to similar operations in conventional systems. This is demonstrated with a set of microbe ...

10 A faster UDP

Craig Partridge, Stephen Pink

August 1993 **IEEE/ACM Transactions on Networking (TON)**, Volume 1 Issue 4

Publisher: IEEE Press

Full text available: pdf(1.29 MB) Additional Information: full citation, references, citings, index terms, review

11 An efficient implementation of SELF a dynamically-typed object-oriented language based on prototypes

C. Chambers, D. Ungar, E. Lee

September 1989 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications OOPSLA '89**, Volume 24 Issue 10

Publisher: ACM Press

Full text available: pdf(2.41 MB) Additional Information: full citation, abstract, references, citings, index terms

We have developed and implemented techniques that double the performance of dynamically-typed object-oriented languages. Our SELF implementation runs twice as fast as the fastest Smalltalk implementation, despite SELF's lack of classes and explicit variables. To compensate for the absence of classes, our system uses implementation-level maps to transparently group objects cloned from the same prototype, providing data type information and eliminating the apparent ...

12 Automatic testing equivalence verification of spi calculus specifications

Luca Durante, Riccardo Sisto, Adriano Valenzano

April 2003 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 12 Issue 2

Publisher: ACM Press

Full text available: pdf(829.73 KB) Additional Information: full citation, abstract, references, index terms

Testing equivalence is a powerful means for expressing the security properties of cryptographic protocols, but its formal verification is a difficult task because of the quantification over contexts on which it is based. Previous articles have provided insights into using theorem-proving for the verification of testing equivalence of spi calculus specifications. This article addresses the same verification problem, but uses a state exploration approach. The verification technique is based on the ...

Keywords: Cryptographic protocols, equivalence verification, state space exploration

13 Selection conditions in main memory

Kenneth A. Ross

March 2004 **ACM Transactions on Database Systems (TODS)**, Volume 29 Issue 1

Publisher: ACM Press

Full text available: pdf(296.54 KB) Additional Information: full citation, abstract, references, index terms

We consider the fundamental operation of applying a compound filtering condition to a set of records. With large main memories available cheaply, systems may choose to keep the data entirely in main memory, in order to improve query and/or update performance.The design of a data-intensive algorithm in main memory needs to take into account the architectural characteristics of modern processors, just as a disk-based method needs to consider the physical characteristics of disk devices. An importa ...

Keywords: Branch misprediction

14 Using cache line coloring to perform aggressive procedure inlining

Hakan Aydin, David Kaeli
March 2000 **ACM SIGARCH Computer Architecture News**, Volume 28 Issue 1
Publisher: ACM Press
Full text available: pdf(701.54 KB) Additional Information: full citation, abstract, index terms

Memory hierarchy performance has always been an important issue in computer architecture design. The likelihood of a bottleneck in the memory hierarchy is increasing, as improvements in microprocessor performance continue to outpace those made in the memory system. As a result, effective utilization of cache memories is essential in today's architectures.The nature of procedural software poses visibility problems when attempting to perform program optimization. One approach to increasing visibil ...

15 The hierarchical simulation language HSL: a versatile tool for process-oriented simulation

D. P. Sanderson, R. Sharma, R. Rozin, S. Treu
April 1991 **ACM Transactions on Modeling and Computer Simulation (TOMACS)**, Volume 1 Issue 2
Publisher: ACM Press
Full text available: pdf(2.68 MB) Additional Information: full citation, references, citings, index terms, review

Keywords: C++, HSL, hierarchy, inheritance, interpreter, modularity, process, simulation programming language

16 Using shape analysis to reduce finite-state models of concurrent Java programs

James C. Corbett
January 2000 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 9 Issue 1
Publisher: ACM Press
Full text available: pdf(284.92 KB) Additional Information: full citation, abstract, references, citings, index terms

Finite-state verification (e.g., model checking) provides a powerful means to detect concurrency errors, which are often subtle and difficult to reproduce. Nevertheless, widespread use of this technology by developers is unlikely until tools provide automated support for extracting the required finite-state models directly from program source. Unfortunately, the dynamic features of modern languages such as Java complicate the construction of compact finite-state models for verification. I ...

Keywords: Java, concurrent systems, finite-state verification, model extraction, modeling, shape analysis, state-space reductions

17 ATUM: a new technique for capturing address traces using microcode

A. Agarwal, R. L. Sites, M. Horowitz
June 1986 **ACM SIGARCH Computer Architecture News , Proceedings of the 13th annual international symposium on Computer architecture ISCA '86**, Volume 14 Issue 2
Publisher: IEEE Computer Society Press, ACM Press
Full text available: pdf(894.10 KB) Additional Information: full citation, abstract, references, citings, index terms

Trace-driven simulation is often used in the design of computer systems, especially

caches and translation lookaside buffers. Capturing address traces to drive such simulations has been problematic, often involving 1000:1 software overhead to trace a target workload, and/or mechanisms that cause significant distortions in the recorded data. A new technique for capturing address traces has been developed to use a processor's microcode to record addresses in a reserved part of main memory as ...

18 Research session 4: query processing and optimization I: Conjunctive selection conditions in main memory

Kenneth A. Ross

June 2002 **Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems**

Publisher: ACM Press

Full text available: pdf(221.81 KB) Additional Information: full citation, abstract, references, citings, index terms

We consider the fundamental operation of applying a conjunction of selection conditions to a set of records. With large main memories available cheaply, systems may choose to keep the data entirely in main memory, in order to improve query and/or update performance.The design of a data-intensive algorithm in main memory needs to take into account the architectural characteristics of modern processors, just as a disk-based method needs to consider the physical characteristics of disk devices. An ...

19 Validation and verification: Local model checking of Java bytecode

Antonella Santone, Gigliola Vaglini

July 2002 **Proceedings of the 14th international conference on Software engineering and knowledge engineering SEKE '02**

Publisher: ACM Press

Full text available: pdf(262.37 KB) Additional Information: full citation, abstract, references, citings

We present a model checking procedure to verify a subset of the Java Virtual Machine Language. The procedure works on a finite tractable state representation of the program: in fact, abstraction techniques are employed, driven by the temporal logic formula representing the property to be checked. A tableau-based method is developed to proof the satisfaction of the formulae: thus not all the program computations are checked, but only those ones interested in the goal of the property verification.

Keywords: model checking, software systems, tableau system, temporal logic

20 Technical papers: Guidelines for interdomain traffic engineering

Nick Feamster, Jay Borkenhagen, Jennifer Rexford

October 2003 **ACM SIGCOMM Computer Communication Review**, Volume 33 Issue 5

Publisher: ACM Press

Full text available: pdf(705.89 KB) Additional Information: full citation, abstract, references, citings, index terms

Network operators must have control over the flow of traffic into, out of, and across their networks. However, the Border Gateway Protocol (BGP) does not facilitate common traffic engineering tasks, such as balancing load across multiple links to a neighboring AS or directing traffic to a different neighbor. Solving these problems is difficult because the number of possible changes to routing policies is too large to exhaustively test all possibilities, some changes in routing policy can have an ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next